Regular Paper

Effective Mobile Search Using Element-based Retrieval

Atsushi Keyaki[1,a)] Jun Miyazaki[1,b)] Kenji Hatano[2,c)]

Received: March 1, 2017, Accepted: July 6, 2017

Abstract: We applied an element-based retrieval approach in this study to improve search accuracy for information retrieval on small screen devices. Finer-grained retrieval units compared with document granules are expected in mobile information retrieval because of limited screen size. An information retrieval system employing information units (*iUnits*), i.e., text fragments that are relevant to a query and atomic in interpreting information, is one of the solutions to finer-grained retrieval units. An iUnit-based mobile information retrieval task, called MobileClick, assumes two-layered information access where the most important iUnits are arranged in the first layer and detailed iUnits for individual intents are arranged in the second layer. This two-layered information access enabled useful information to be presented with fewer page transitions. We first adopt element-based retrieval to identify elements containing relevant descriptions in a document to tackle MobileClick. The next step is scoring the iUnits where we assigned higher scores to iUnits that are similar to elements with higher scores. Our experimental evaluations demonstrated that our method is more accurate than a baseline by 9.7%.

Keywords: mobile information retrieval, element-based retrieval, MobileClick, iUnit

1. Introduction

Google has reported that the number of Web searches from mobile devices is superior to that from PCs in 10 countries including the USA and Japan [*1]. The fact that the number of Web searches from mobile devices is growing indicates appropriate information access methodology is being materialized for mobile information retrieval (IR), which is crucial and urgent. Information selection for mobile IR is greatly required because mobile devices generally have a small screen mounted to them and simultaneously display limited amounts of information. In other words, document-granular IR is not effective for mobile IR.

MobileClick [9] is a task conducted at the NII Testbeds and Community for Information access Research (NTCIR) workshop. The main aim of MobileClick is to present relevant search results in mobile IR. When a query is issued, two-layered search results are displayed to enable direct and immediate information acquisition without unnecessary page/site transitions in MobileClick. **Figure 1** has an example of two-layered search results. The first layer is comprised of important and general information to a query and links to a details page (the second layer) of the intents of the query. Thus, the two-layered search results are expected to provide effective mobile IR even with limited screen sizes. The retrieval units of MobileClick are information units (*iUnits*). The definition of iUnits is text fragments that are relevant to a query and atomic in interpreting information [*2]. An iUnit is composed of arbitrary granular information, such as a few terms, a phrase, and a sentence. General information in the first layer and details

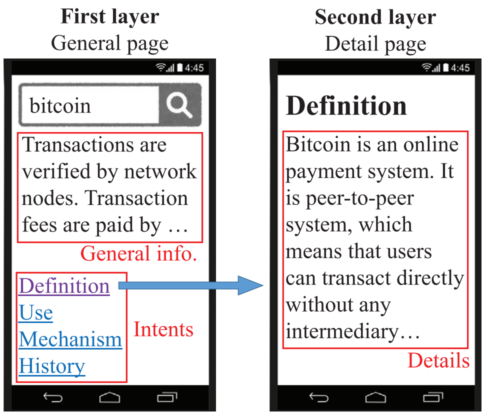

First layer
General page
bitcoin
Transactions are verified by network nodes. Transaction fees are paid by …
General info.
Definition
Use
Mechanism
History
Intents
Second layer
Detail page
Definition
Bitcoin is an online payment system. It is peer-to-peer system, which means that users can transact directly without any intermediary…
Details

Fig. 1 Two-layered search results.

in the second layer contain multiple iUnits that are ranked by importance.

MobileClick has been worked through with various approaches [2], [3], [4], [5], [15], [19], [27]. Apart from these, we employed element-based retrieval [6], [14]. The retrieval units of element-based retrieval are "*elements*" in structured documents, such as those in Extensible Markup Language (XML) and HyperText Markup Language (HTML). More specifically, each concatenated text in an element node (plain text between a pair of start and end tags) is treated as a search target. This means element-based retrieval involves finer-grained IR as well as MobileClick. Document structures are leveraged in element-based retrieval to identify the most appropriate elements that satisfy users' information needs. We assume that both MobileClick

1 Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
2 Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
a) keyaki@lsc.cs.titech.ac.jp
b) miyazaki@cs.titech.ac.jp
c) khatano@mail.doshisha.ac.jp

This paper is an extended version of our NTCIR-12 Ref. [12].
*1 https://adwords.googleblog.com/2015/05/building-for-next-moment.html (accessed: December 3rd, 2016)
*2 Although Tajima et al. [25] defined an iUnit as a subgraph containing all distinct query keywords, this condition is not imposed in MobileClick.

and element-based retrieval aspire toward the same direction and many of their findings could be diverted to each other, although their retrieval units are not exactly the same.

We hypothesize the following in this study:

Hypothesis. *Relevant descriptions,* i.e.*, important iUnits are contained in elements highly scored by element-based retrieval.*

We will now describe the outline of our proposal based on this hypothesis. The first step is scoring elements with an existing element-based retrieval technique to generate a ranked element list per query. The second step is generating two-layered search results with the ranked list. An intuitive explanation of our method is that higher scores are assigned to iUnits that are similar to elements with higher scores.

This paper is structured as follows. Overviews of MobileClick and element-based retrieval are reported in Sections 2 and 3. Next, we propose a method of generating accurate two-layered search results with element-based retrieval in Section 4, followed by its experimental evaluation in Section 5. We then review related studies in Section 6. Finally, Section 7 concludes the paper.

2. Overview of MobileClick

MobileClick is descended from 1CLICK-1 [24] at NTCIR-9 and 1CLICK-2 [7] at NTCIR-10. Then, MobileClick was dealt with in MobileClick-1 [8] at NTCIR-11 and MobileClick-2 [9] at NTCIR-12. MobileClick-1 and MobileClick-2 are not exactly the same tasks since data provided by task organizers are different. MobileClick implies MobileClick-2 in this paper, unless otherwise stated. For more details, refer to MobileClick-2 overview Ref. [9].

MobileClick is comprised of two steps, i.e., Step 1) ranking iUnits and Step 2) constructing two-layered search results using ranked iUnits. Steps 1) and 2) are tackled as an *iUnit Ranking* subtask for the former and an *iUnit Summarization* subtask for the latter.

2.1 Test Collection

The data provided by the task organizers are below:

Query One hundred ambiguous or underspecified queries are sampled by the task organizers from query logs of a Web search system. Queries are classified into four categories; 20 CELEBRITY queries (names of celebrities such as artists and actors), 20 LOCAL queries (landmarks and facilities), 40 DEFINITION queries (ambiguous terms such as "bitcoin," "euro," and "smartphone"), and 20 QA queries (natural language questions).

Document set At most the top 500 documents per query are gathered with a Web search system, Bing [*3].

iUnit The definition of an iUnit is a text fragment that is relevant to a query and that is atomic in interpreting information. Thus, an iUnit is composed of arbitrary granular information, such as a few terms, a phrase, and a sentence. For example, the iUnits of a query "*christopher nolan*" are "*runs the production company 'Syncopy Inc.'*," "*married to a film producer, Emma Thomas*," and "*Movie: 'Interstellar'*." The provided iUnits are extensively extracted from the document set by assessors hired by organizers.

[*3] http://www.bing.com/

Intent The notion of "*intent*" is introduced from NTCIR tasks INTENT [23] and IMine [26]. An intent represents a specific interpretation of an ambiguous query or an aspect of a faceted query. As examples, the intents of a query "*jaguar*" are "*Car*" and "*Animal*," while the intents of a query "*real Madrid*" are "*Players*," "*Achievement*," and "*History*." Intents in MobileClick are used to evaluate the importance of iUnits and used as links to second layers in iUnit Summarization subtasks. Each intent has its importance ("*intent probability*"). An intent probability of a query is defined as the ratio of voters who agreed to become interested in the intent when the query is issued.

Importance of iUnits The importance of iUnits is used in evaluating the accuracy of a method. iUnit importance is assessed by each intent (*per-intent importance*), because a certain iUnit can be important for an intent, although this iUnit may not be important for other intents. The *global importance*, $G(u)$, of an iUnit, u, is derived from the per-intent importance of the iUnit and the intent probabilities as:

$$G(u) = \sum_{i \in I_q} P(i|q) g_i(u) \tag{1}$$

where I_q is a set of intents for query q, i is an intent in I_q, $P(i|q)$ is the intent probability of q, and g_i is the per-intent importance of u in terms of i. This means an iUnit with a high scores in terms of an intent can have low global importance when the iUnit is not important for other intents. Conversely, the global importance of an iUnit without a high score in terms of any intent can be high when the iUnit is important to some extent for many intents.

Note that each data item contains Japanese and English data. We used English data in this study.

2.2 iUnit Ranking Subtask

Before two-layered search results are constructed, iUnits need to be ranked by importance. The importance of each iUnit in the ranking subtask is calculated by global importance.

Evaluation measures for the iUnit Ranking subtask are $nDCG_k$ (k = 3, 5, 10, and 20) [1] and the Q-measure [21]. Metric normalized discounted cumulative gain (nDCG) is commonly used as a measure to evaluate document retrieval. The Q-measure is a recall-based graded-relevance measure, while nDCG is a rank-based graded-relevance measure. In other words, Q-measure is used to not only evaluate the performance of top-k iUnits but the overall performance of a list. Thus, Q-measure is defined as a formal measure and effectiveness of a system is determined by Q-measure score.

Metric nDCG is calculated as:

$$DCG_K = \sum_{k=1}^{K} \frac{G(u_k)}{\log_2(k+1)} \tag{2}$$

$$nDCG_K = \frac{DCG_K}{iDCG_K} \tag{3}$$

where u_k is the k-th iUnit and $G(u_k)$ is the global importance of

u_k. The *iDCG* is the *DCG* of an ideal ranked list and works as normalization.

The Q-measure is calculated with Eq. (4).

$$Q = \frac{1}{R}\sum_{r=1}^{M} IsRel(u_r)\frac{\sum_{r'=1}^{r}(G(u_{r'}) + IsRel(u_{r'}))}{\sum_{r'=1}^{r} G(u^*_{r'}) + r} \tag{4}$$

where R is the number of iUnits with non-zero global importance, M is the length of the ranked list, and $u^*_{r'}$ is the r-th iUnit of an ideal ranked list.

2.3 iUnit Summarization Subtask

The search result format for the iUnit summarization subtask is composed of two-layered search results where the first layer is comprised of iUnits with the most important and general information and intents, and the second layer linked by the intents is comprised of iUnits, whose details in terms of each intent are depicted in Fig. 6. Each of the first and second layers is comprised of ranked iUnits gained in the iUnit ranking. This format helps users to aquire information they need with fewer page transitions.

Each layer can only present at most 420 characters in the iUnit summarization subtask because the screen size of mobile devices is limited. Symbols and white spaces are not counted and the importance of an iUnit in the second layer become zero when the iUnit appears both in the first and the second layers.

An evaluation measure of the iUnit summarization subtask, M-measure, is designed based on the user model as follows:

- A user becomes interested in only one intent based on intent probability $P(i|q)$.
- A user follows a rule in reading search results described below:
  (1) A user starts to read search results from the beginning of the first layer and continues to read until L characters, excluding symbols and white spaces.
  (2) A user moves to and reads the search results of the second layer, s_j, when he/she has reached the end of a link, l_i, if he/she becomes interested in the intent.
  (3) After reading the search results of the second layer, s_j, a user resumes reading the search results of the first layer from the end of the link, l_j.

Note that there are possible text trails to the number of intents because a user only becomes interested in one intent. The information gain of each text trail is measured by using the U-measure [22], where the offsets of iUnits are taken into account. More precisely, the U-measure increases when more important iUnits are arranged in former positions of a text trail. Then, the *expected* information gain of all text trails is measured by using the M-measure where a user probabilistically reads search results (text trails).

The M-measure based on the discussion above is calculated as:

$$M = \sum_{i \in I_q} P(i|q)U_i(t_i) \tag{5}$$

where I_q is a set of intents for query q, i is an intent in I_q, $P(i|q)$ is the intent probability of intent i for q, and $U_i(t_i)$ is a value of the U-measure of i's text trail t_i. The U-measure is calculated with Eqs. (6), (7), and (8).

$$U_i(t) = \sum_{j=1}^{|t|} g_i(u_j)d(u_j) \tag{6}$$

$$d(u) = \max(0, 1 - \frac{pos_t(u)}{L}) \tag{7}$$

$$pos_t(u) = \sum_{j'=1}^{j} chars(u_{j'}) \tag{8}$$

where $g_i(u_i)$ is the importance of iUnit u in terms of i, d is the position-based decay function, and $chars(u)$ is the number of characters in i that excludes symbols and white spaces. Hence, the iUnits in the first layer are evaluated with global importance and the iUnits in the second layer are evaluated with importance in terms of each intent.

3. Overview of Element-based Retrieval

This section overviews element-based retrieval. For more details, refer to overview paper of element retrieval [6].

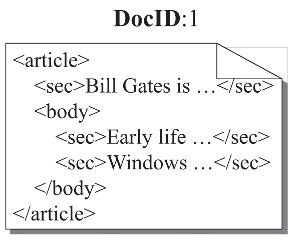
3.1 Main Objective of Element-based Retrieval

The main objective of element-based retrieval lies in identifying only and all relevant descriptions to a query, namely, relevant elements, and ranking them in descending order of relevance. Element-based retrieval is aimed at reducing user labor in information retrieval activities, since users do not have to find relevant descriptions by themselves.

3.2 Element and Text Overlaps

We have presented concrete examples in **Figs. 2**, **3**, and **4** to explain the definitions of elements of structured documents such as XML and HTML.

Figure 2 illustrates an XML document of DocID 1. Figure 3 depicts a tree that has been translated from Fig. 2. Structured documents can be expressed as a tree, which helps to easily identify document structures. The pair of start and end tags in Fig. 2 represents an element node in the tree in Fig. 3, and the nested structure of elements represents an ancestor-descendant relationship. An


DocID:1

```
<article>
  <sec>Bill Gates is …</sec>
  <body>
    <sec>Early life …</sec>
    <sec>Windows …</sec>
  </body>
</article>
```

Fig. 2 XML document.

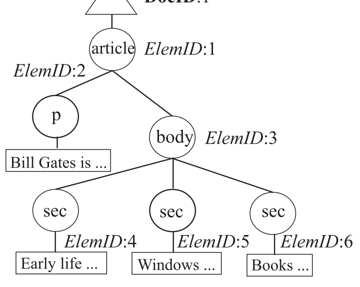


Fig. 3 Representation in tree.

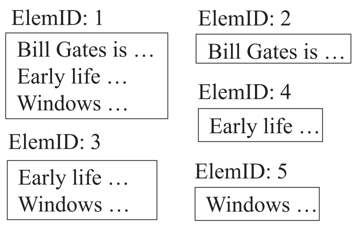


Fig. 4 Elements.

ElemID is assigned to each element in document order. Each element in Fig. 4 corresponds to a text composed of text nodes in the tree in Fig. 3. Thus, an element that corresponds to an `article` node has the concatenated text of all text nodes in the tree. Similarly, an element that corresponds to a `body` node has the concatenated text of text nodes in the three `sec` nodes. This demonstrates why text overlaps occur between ancestor-descendant elements in the same structured document. An ancestor element and a descendant element are expressed as a larger-granular element for the former and a smaller-granular element for the latter [*4].

Let us suppose that a user is seeking information about "Early life . . . ," "Windows . . . " and "Books · · · ." An element-based retrieval system tries to extract the element of ElemID 3 because the element contains all the information that the user needs and no further information.

3.3 Search Results for Element-based Retrieval

There are some approaches in constructing a ranked list as search results. Some early researchers did not take into account text overlaps and just constructed a list where elements were arranged in descending order of relevance. We have called the list an "*overlapping list*" in this paper. After Kazai et al. [10] reported that text overlaps adversely affect search accuracy, many researchers remove text overlaps by eliminating duplicate elements [*5]. Consequently, "*non-overlapping lists*" is constructed as search results. The largest element-based retrieval project of the INitiative for the Evaluation of XML (INEX) [*6] retrieval also employs a non-overlapping list [6]. In this project, at most 1,500 elements can be proposed for each query.

We use three approaches in constructing a non-overlapping list:

OneElem Only one element, i.e., the element with the highest score, can be extracted from each document. In other words, the maximum number of search results (elements) is equal to that of documents containing a query keyword. Given an overlapping list gained from a certain query as shown in **Table 1**, only the element of which ElemID is 5 ($elem_5$ for short) is extracted and contained in a non-overlapping list.

MultiElem Multiple elements from each document can be extracted in descending order of relevance if no text overlaps occur. The maximum number of search results is more than that of the document containing a query keyword. We demonstrate the process of generating a non-overlapping list from the overlapping list in Table 1. First, $elem_5$ is extracted. Then, $elem_2$ is successfully extracted since no overlaps occur. Extraction of $elem_3$ fails because $elem_5$, a descendant of $elem_3$, has already been extracted. Similarly, $elem_1$ cannot be extracted. Extraction is completed by extraction of $elem_6$. Consequently, a non-overlapping list is composed of $elem_5$, $elem_2$, and $elem_6$.

WholeDoc For each document, the largest granular element, i.e., a whole document is extracted. Some of relevant descriptions in a document may not be extracted with two former approaches. Conversely, this approach ensures all relevant descriptions in a document are extracted. The problem is that non-relevant descriptions are also extracted because a document generally contains both relevant and irrelevant descriptions. Thus, this approach deviates from the original objective of element-based retrieval. The number of the maximum search results (elements) is equal to that of documents containing a query keyword. In this approach, $elem_1$ is extracted and a non-overlapping list is composed of only $elem_1$.

[*4] Precisely this is element overlap, however, this overlap is commonly expressed as text overlap. Moreover, in this paper, it is not expressed as text overlaps when elements not having ancestor-descendant relationships contain the same text.

[*5] We remove a whole element when text overlaps occur. In other word, we do not remove duplicate text from overlapping elements.

[*6] http://inex.mmci.uni-saarland.de/

Table 1 Overlapping list.

ElemID	Score
5	.9
2	.6
3	.5
1	.4
6	.2

3.4 Accurate Element-based Retrieval

Most scoring functions for element-based retrieval are derived from those for document retrieval. Thus, some statistics used in these scoring functions are common. The main difference between these scoring functions is that the notion of "*attributes*" of elements is taken into account in element-based retrieval. More concretely, global statistics for element-based retrieval are calculated with elements with the same attributes, while global statistics for document retrieval are calculated with all documents. Some approaches to the classification of element attributes have been proposed. Typical approaches are tag-based classification where elements with the same tags have the same attribute and path expression-based classifications where elements with the same path expressions have the same attributes.

Major scoring functions for element-based retrieval are term frequency-inverse path frequency (TF-IPF) [16], BM25E [17], and the query likelihood model for element-based retrieval [18]. We employed BM25E in this study, because it has been reported to be the most accurate [13]. A weight, $w(a, e, t)$, of a term, t, in an element, e, with an attribute, a, is calculated in BM25E as:

$$w(a, e, t) = \frac{(k_1 + 1)tf_{e,t}}{k_1((1 - b) + b\frac{el_e}{avel_a}) + tf_{e,t}} \cdot \log \frac{N_a - af_{a,t} + 0.5}{pf_{a,t} + 0.5} \quad (9)$$

where $tf_{e,t}$ is the term frequency of t in e, el_e is the length of e (the number of terms in e), and $avel_a$ is the average length of elements with a. Here, N_a is the number of elements with a, $af_{a,t}$ is the number of elements with a containing t, and k_1 and b are parameters. We set commonly used values for parameters, i.e., $k_1 = 2.5$ and $b = 0.85$.

Although it has been reported that path expression-based attribute classification is the most accurate for the INEX Wikipedia test collection [6], our past investigation [14] found that global statistics could not be accurately calculated with the path expression-based attribute classification when there are insufficient numbers of elements with attributes. The number of document sets we used in experimental evaluations is not large, as we explained in Section 2.1. Hence, we apply the tag-based attribute classification in this study, viz., a in Eq. (9) was set as a tag.

Finally, a score, $Score(e)$, of element e is computed with:

$$Score(e) = \sum_{t_i \in T} w(p, e, t_i) \tag{10}$$

where T is a set of query keywords.

4. Proposed Method

Both important iUnits and relevant elements represent relevant descriptions, which is a summary of Sections 2 and 3. Since an accurate element-based retrieval technique has already existed, we focused on identifying important iUnits using elements expected to be relevant. Thus, we propose a method based on Hypothesis: "Relevant descriptions, i.e., important iUnits are contained in elements highly scored by element-based retrieval."

Our method of the iUnit Ranking and iUnit Summarization subtasks will be discussed in what follows. Note that the proposed method is an extension of naive one [12] presented at NTCIR-12 conference.

4.1 Proposal for iUnit Ranking Subtask

iUnits provided by the task organizers are ranked by importance in the iUnit Ranking subtask.

Figure 5 illustrates the procedure for the proposed method. First, elements are extracted from a HTML document set. Next, a ranked list is constructed where elements are ordered by relevance using element-based retrieval. Then, iUnits are ranked by calculating similarities between elements and iUnits. Higher scores are assigned in this step to iUnits that are similar to highly ranked elements. Therefore, a score, $Score(u)$, of iUnit u is calculated as:

$$Score(u) = \sum_{e \in E} \frac{sim(u, e)}{decay(e)} \tag{11}$$

where E is a set of elements used in iUnit scoring (elements containing a query keyword), e is an element contained in E, and

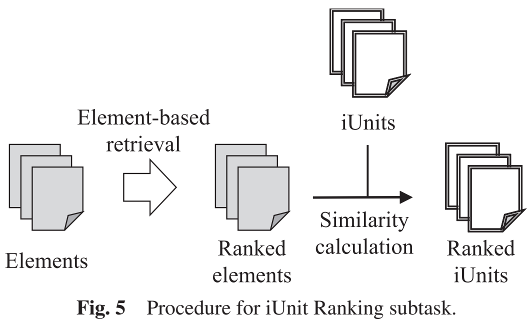


Fig. 5 Procedure for iUnit Ranking subtask.

$sim(u, e)$ is similarity between u and e. Here, $decay(e)$ works as a decay function using e's rank order.

We will next describe methods of similarity calculation $sim(u, w)$, decay function of rank order $decay(e)$, and selection of E.

Since no appropriate method of calculating similarities between an arbitrary iUnit and an element is decisive, we examined three methods of calculating similarities, i.e., $Freq$ (12), *Ratio* (13), and *Jaccard* (14).

$$sim_{Freq}(u, e) = count(u, e) \tag{12}$$

where $count(u, e)$ is the number of terms that co-occurred in both an iUnit, u, and an element, e.

$$sim_{Ratio}(u, e) = \frac{count(u, e)}{length(u)} \tag{13}$$

where $length(u)$ is the number of terms contained in u.

$$sim_{Jaccard}(u, e) = \frac{|u \cap e|}{|u \cup e|} \tag{14}$$

We propose two variations of decay functions, i.e., *Rank* (15) and *LogRank* (16), to emphasize the effect of highly ranked elements.

$$decay_{Rank}(e) = rank(e) \tag{15}$$

where $rank(e)$ is the rank order of e.

$$decay_{LogRank}(e) = 1 + \log(rank(e)) \tag{16}$$

The main objective of *LogRank* is moderating a decay rate as rank order increases.

We consider three variations for the selection of E used in iUnit scoring:

$\boldsymbol{E_{AllElem}}$ All elements in a ranked list are used.

$\boldsymbol{E_{TopPerElem}}$ Elements of top-k percentage are used.

$\boldsymbol{E_{TopElem}}$ Top-k elements are used.

4.2 Proposal for iUnit Summarization

Two-layered search results are constructed for the iUnit Summarization subtask. We explore appropriate iUnit ranking methods for each layer and the intent ranking method. Then, we employ the baseline approach in arranging iUnits and intents in the first layer [8].

We concretely show our approach to arrangement in **Fig. 6**.

(1) All intents are arranged from the end of the first layer.

(2) iUnits are arranged from the beginning of the first layer unless the number of characters extracted exceeds the threshold.

(3) iUnits in the second layer are arranged for individual intents unless they exceed the threshold. Note that iUnits that have already been arranged in the first layers will not be arranged in the second layer.

We will discuss intent and iUnit rankings in the following subsections.

4.2.1 Intent Ranking

More accurate search results can be presented when higher

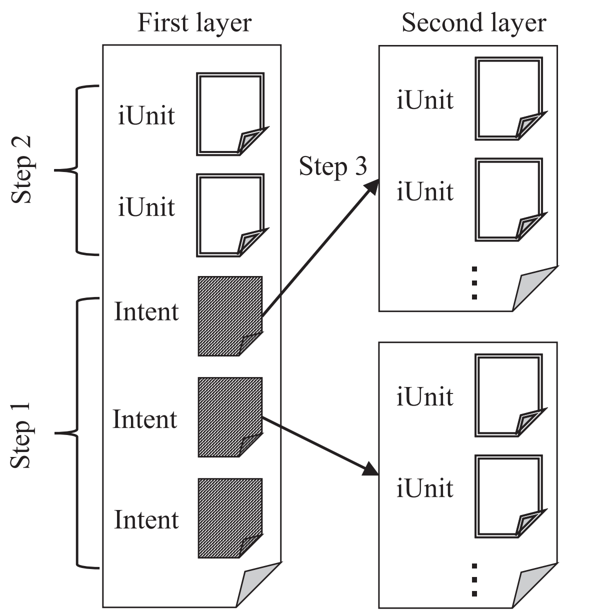


Fig. 6 Construction of search results for the iUnit Summarization subtask.

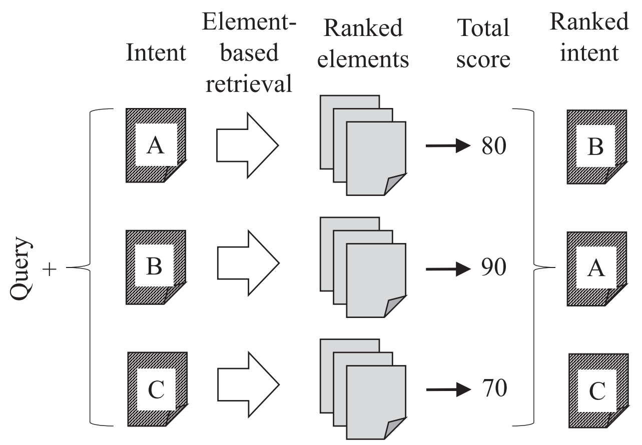


Fig. 7 The overview of SumElemScore.

intent probabilities are arranged in higher rank order, since the evaluation measure, M-measure, takes into account the position-based decay function.

The proposed method of intent ranking aims at assigning higher scores to intents expected to gain more information. We considered three methods:

SumElemScore An expanded query where each intent added to an original query is used for element-based retrieval to gain scored elements. Then, an intent score is calculated by using the sum of element scores of the scored elements. There is an overview in **Fig. 7**. The intent score of an intent i, $S_{Int}(i)$, is calculated as follows:

$$S_{Int}(i) = \sum_{e \in E} S_{EL}(e, q_i) \tag{17}$$

where E is a set of scored elements, e is an element in E, $S_{EL}(e, i)$ is a score of e queried with a original query and i.

NumResultElem The number of scored elements with an expanded query becomes an intent score as flows:

$$S_{Int}(i) = N_{EL}(i) \tag{18}$$

where $N_{EL}(i)$ is the number of elements containing an expanded query of which intent is i.

NumWebSearch Both the two previous methods are based on statistics given from an inner document set. It is possible that these statistics stand for limited information because there are at most 500 documents per query. Hence, we decided to acquire outer information from the document set. We issue an expanded query to a Web search system [*7] and the number of search results becomes an intent score.

$$S_{Int}(i) = N_{Web}(i) \tag{19}$$

where $N_{EL}(i)$ is the number of Web search results when issued an expanded query of which intent is i.

4.2.2 Query for iUnit Ranking

Preferable iUnits to be arranged at the first layer are important iUnits for many intents. Keeping this in mind, we considered two iUnit ranking approaches:

AllIntent A full expanded query where all intents are added to an original query is used for element-based retrieval to gain scored elements.

EachIntent Expanded queries are issued and each element score is calculated by using the sum of element scores of all expended queries.

The expected condition of iUnits in the second layer is dedicated to an intent. Two approaches were evaluated where:

ExpandedQuery An expanded query is used for element-based retrieval.

IntentQuery An intent without an original query is used as an intent query for element-based retrieval to emphasize the intent itself.

5. Experimental Evaluation

This section explains how we verified the effectiveness of the proposed method for the subtasks of iUnit Ranking and iUnit Summarization. We used the data referred to in Section 2 and BM25E as an element-based retrieval scoring function introduced in Section 3.4.

5.1 Evaluation of iUnit Ranking Subtask

We discussed the iUnit ranking function in Eq. (11), methods of calculating similarities ($sim(u, w)$ in Eq. (11)), decay function of rank order ($decay(e)$ in Eq. (11)), and selection of elements used in iUnit scoring (E in Eq. (11)) in Section 4.1. All their variations were evaluated and compared. The formal evaluation measure is the Q-measure. Among variations, *LogRank*, *TopPerElem*, and *TopElem* require a parameter. To set a parameter, we conducted two-fold cross-validation for avoiding overfitting. Specifically, we classified queries into two groups where one group comprises queries with odd ID (*odd queries*) and the other group comprises query with even ID (*even queries*). One group is used to tune a parameter and then the other group is evaluated with the parameter. Eventual evaluation result is defined as the average of two groups. Note that the best parameters vary for each test collection.

[*7] We used Bing as well as the engine used for collecting the document set.

Table 2 Evaluation results for iUnit Ranking subtask: We found the best settings were using $sim_{Ratio}(u, e)$ for the similarity calculations, $decay_{Rank}(e)$ for decay function using rank order, and $E_{TopPerElem}$ for selection of elements used in iUnit scoring. A non-overlapping list is more accurate than an overlapping list. We found the accuracy of element-based retrieval was worse than that of document retrieval.

	$nDCG_3$	$nDCG_5$	$nDCG_{10}$	$nDCG_{20}$	Q-measures
Freq	.7237	.7510	.7966	.8654	.8947
Ratio	.7325	.7499	.7995	.8707	.8969
Jaccard	.6849	.7192	.7717	.8511	.8800
Rank	.7443	.7632	.8060	.8748	.9005
LogRank	.7341	.7518	.8017	.8727	.8983
TopPerElem	.7453	.7627	.8065	.8751	.9007
TopElem	.7437	.7614	.8045	.8743	.9001
AllIntent	.7551	.7712	.8104	.8796	.9031
EachIntent	.7511	.7688	.8112	.8776	.9025
OneElem	.7641	.7764	.8140	.8815	.9047
MultiElem	.7604	.7742	.8132	.8806	.9044
WholeDoc	**.7613**	**.7750**	**.8154**	**.8815**	**.9050**

Table 3 Tuning of *LogRank* with Q-measure.

	Odd queries	Even queries
$\log_{1.1}$	.8906	**.9048**
$\log_{1.5}$	.8917	.9047
$\log_2$	**.8918**	**.9048**
$\log_e$	.8918	.9034
$\log_{10}$	.8918	.9032
Tuned parameter	**.8918**	**.9048**

We have used an overlapping list for the results of element-based retrieval unless otherwise stated, although a non-overlapping list is generally used, as we explained in Section 3.3. This is because we had no idea on what kind of list was appropriate for this situation.

5.1.1 Results from Evaluation of Proposed Method

The results we obtained from evaluation are listed in **Table 2**.

First, let us focus on the methods of calculating similarities, i.e., *Freq*, *Ratio*, and *Jaccard*. Any decay function is not applied to elements ($decay(e) = 1$) and all elements were used in iUnit scoring ($E_{AllElem}$). Out of the three methods of calculating similarities, *Ratio* was the most accurate.

We will next explain our evaluation of decay functions of rank order. We evaluated *Rank* and *LogRank* with $sim_{Ratio}(u, e)$ and $E_{AllElem}$. We varied the base for *LogRank* between 1.1, 1.5, 2, the Napierian logarithm, and 10 to find the best value [*8]. As summarized in **Table 3**, the best base value of both odd and even queries was two, although rounded values of $\log_{1.1}$ and $\log_2$ for even queries are the same (.9048). Odd queries were evaluated with the parameter tuned with even queries, so as even queries. Then, although both *Rank* and *LogRank* improved accuracy, *Rank* was more accurate.

We moved on to select elements used in scoring iUnits. We explored the best parameters for $E_{TopPerElem}$ and $E_{TopPerElem}$ with $sim_{Ratio}(u, e)$ and $decay_{Rank}$ listed in **Tables 4** and **5**. The parameter, k, represents top elements of the k percentage for *TopPerElem* and top k elements for *TopElem*. The results suggest the top 33% of elements are best for both odd and even queries in terms of *TopPerElem*. In contrast, the top 700 and 400 elements are best for odd queries and even queries in terms of *TopElem*. Accuracy improved by applying element selection *TopPerElem*, although *TopPerElem* deteriorated in contrast. Because the number of elements in a ranked list differs by query, selection based on percentage is more suitable.

We also evaluated all possible combinations, which resulted in $sim_{Ratio}(u, e)$ being the best for similarity calculations, $decay_{Rank}(e)$ for decay function of rank order, and $E_{TopPerElem}$ for selection of elements used in iUnit scoring, i.e., *TopPerElem*. We therefore used *TopPerElem* as the method of iUnit ranking in the experiments that are described in the following.

5.1.2 Further Investigations

We evaluated *AllIntent* and *EachIntent* that were proposed for the iUnit Summarization subtask to confirm their potential. Although both methods increased accuracy, *AllIntent* was better [*9].

We next examined the effect of using a non-overlapping list on iUnit ranking (the ranking method was *AllIntent*). We found that all approaches for the three non-overlapping lists: *OneElem*, *MultiElem*, and *WholeDoc* described in Section 3.3, outperformed the overlapping list. The document retrieval approach *WholeDoc* was unexpectedly more accurate than element-based approaches *OneElem* and *MultiElem*.

Our previous survey [11] on the document structure of Web documents suggested that physical document structures do not agree with logical document structures such as chapters, clauses, and sections. This means element-based retrieval may fail to exert full influence with unprocessed Web documents. Thus, the accuracy of *OneElem* and *MultiElem* are expected to be improved by data trimming of Web documents.

Finally, we compared the results obtained with our methods with formal run results from other iUnit Ranking subtask participants [*10], as listed in **Table 6** [*11]. Improvements in the table stand for improved percentages compared with the baseline (*ORG-L*).

Table 4 Best parameters for *TopPerElem* with Q-measure.

Parameter k	Odd queries	Even queries
3	.8918	.9031
5	.8927	.9039
10	.8937	.9048
20	.8950	.9053
25	.8950	.9059
33	**.8951**	**.9063**
50	.8946	.9055
Tuned parameter	**.8951**	**.9063**

Table 5 Best parameters for *TopElem* with Q-measure.

Parameter k	Odd queries	Even queries
40	.8963	.9023
50	.8909	.9028
100	.8929	.9041
400	.8944	**.9061**
500	.8947	.9056
700	**.8949**	.9057
1,000	.8947	.9056
1,500	.8947	.9058
2,000	.8945	.9059
Tuned parameter	.8944	.9057

[*8] We also used the sigmoid function with some slope values. It turned out the logarithm worked best.

[*9] INTENT [23] and IMine [26] work on automatic intent extraction. Although we can apply these to intent extraction, this study exploited given intents.

[*10] Details of important methods are described in Section 6.

[*11] Only the most accurate run is shown for a team who submitted multiple runs. The run results for our team (*titec*) [12] were excluded.

Table 6 Comparison with other iUnit Ranking subtask participants.

Team Is	Q-measures	Improvements
WholeDoc	**.9050**	**0.8%**
OneElem	.9047	0.8%
cuis	.9042	0.7%
IRIT	.9036	0.7%
UHYG	.9028	0.6%
IISR	.9004	0.3%
ORG-L	.8975	–
RISAR	.8972	0.0%
ALICA	.8959	−0.2%
YJST	.8953	−0.2%
ORG-R	.8859	−0.3%
JUNLP	.8859	−0.3%

Table 7 Evaluation results by category for iUnit Ranking subtask.

	CELEBRITY	LOCAL	DEFINITION	QA
WholeDoc	.901 (1.7%)	.896 (0.6%)	.898 (0.6%)	.932 (0.7%)
OneElem	.899 (1.5%)	.897 (0.7%)	.899 (0.7%)	.930 (0.5%)
cuis	.906 (2.3%)	.885 (−0.6%)	.901 (0.9%)	.928 (0.3%)
ORG-L	.886 (–)	.891 (–)	.893 (–)	.926 (–)

Table 8 Evaluated results for iUnit Summarization subtask: The best setting was using *AllIntent* for the first layer, *IntentQuery* for the second layer, and *NumWebSearch* for intent ranking. Experiments with a non-overlapping list indicated element-based retrieval was better than document retrieval.

	M-measures
Simple (TopPerElem)	18.105
AllIntent & ExpandedQuery	18.442
EachIntent & ExpandedQuery	18.315
AllIntent & IntentQuery	18.447
EachIntent & IntentQuery	18.313
SumElemScore	18.428
NumResultElem	18.365
NumWebSearch	18.469
OneElem	**18.530**
MultiElem	18.508
WholeDoc	18.363

Our methods *WholeDoc* and *OneElem* achieved higher accuracy than the state-of-the-art *cuis*. There were no statistically significant differences between our methods and *ORG-L* in the results of sign tests. **Table 7** summarizes the evaluated results by category. The values in brackets represent improvements compared with *ORG-L*. Our methods improved accuracy in all categories compared with *ORG-L*. However, *cuis* was more accurate than our methods in CELEBRITY and DEFINITION categories. The sign tests confirmed that there were no statistically significant differences between any pairs of methods in any categories.

5.2 Evaluation of iUnit Summarization Subtask

Section 4.2 discussed how two-layered search results were constructed for the iUnit Summarization subtask. We provided query candidates for the first and second layers and also intent ranking methods. The M-measure was the formal measure.

5.2.1 Evaluated Results for Proposed Methods

Table 8 lists the accuracy of the proposed methods.

The proposed method *TopPerElem* is the simple method where an original query was used for the first layer, an expanded query (concatenation of an original query and an intent) was used for the second layer, and intent order was random (ordered by intent IDs).

There were four query combinations since we considered *AllIntent* and *EachIntent* for the first layer and *ExpandedQuery* and *IntentQuery* for the second layer. The results indicated the pair of *AllIntent* and *IntentQuery* was the most accurate.

The fact that the combination with *ExpandedQuery* was better than that with *IntentQuery* by focusing on *EachIntent* suggests that iUnits with higher scores in terms of a specific intent are not arranged in the second layer but in the first layer with *EachIntent*. Simultaneously, iUnits with higher scores in terms of multiple intents are not arranged in the first layer. This is why the pair of *AllIntent* and *IntentQuery* was the best. The experiments described in the following employed *AllIntent* and *IntentQuery*.

Intent ranking methods *SumElemScore*, *NumResultElem*, and *NumWebSearch* will be explained. *NumWebSearch* increased accuracy, while *SumElemScore* and *NumResultElem* decreased it. This indicates the number of Web search results can be a useful criterion in identifying an intent that is expected to gain more information. It also indicated that it was difficult to identify informative intent from the statistics gain of a small-scale data set.

As a result of these experiments, we found accuracy was the best with *AllIntent* for the first layer, *IntentQuery* for the second layer, and *NumWebSearch* for intent ranking.

5.2.2 Further Investigations

We conducted experiments with a non-overlapping list as well as the iUnit Ranking subtask with *AllIntent*, *IntentQuery*, and *NumWebSearch*. Consequently, we obtained different trends from that for the iUnit Ranking subtask. The three approaches to the non-overlapping list were more accurate in the order of *OneElem*, *MultiElem*, and *WholeDoc*. This is to say, element-based retrieval was better than document retrieval. We presumed that element-based retrieval would be appropriate for a task where relevant descriptions are extracted with limited characters because the iUnit Ranking subtask ranked all provided iUnits, while the iUnit Summarization subtask extracted at most 420 characters.

We compared our method *OneElem* with formal run results from other iUnit Summarization subtask participants in the final experiments, as listed in **Table 9**[*12]. Improvements in the table stand for improved percentages compared with the baseline (*ORG-T*). *OneElem* attained the highest accuracy. There were statistically significant differences between *OneElem* and the second highest method, *ORG-L*, in the results for sign tests ($p < 0.01$). Additionally, we evaluated *OneElem* and *ORG-T* by category, as shown in **Table 10**. The values in brackets represent improvements compared with *ORG-T*. *OneElem* improved accu-

Table 9 Comparison with other iUnit Summarization subtask participants.

Team IDs	M-measures	Improvements
OneElem	**18.530**	**9.7%**
ORG-T	16.898	–
YJST	16.887	0.0%
IRIT	16.563	−0.2%
cuis	16.419	−0.3%
RISAR	16.047	−0.5%
ORG-R	14.105	−16.5%
UHYG	13.055	−22.7%
JUNLP	11.703	−30.7%
ALICA	8.497	−49.7%

[*12] Only the most accurate run is indicated for a team that submitted multiple runs. The run results for our team (*titec*) [12] were excluded.

Table 10 Evaluated results by category for iUnit Summarization subtask.

	CELEBRITY	LOCAL	DEFINITION	QA
OneElem	30.0 (9.1%)	12.8 (7.6%)	17.6 (12.7%)	14.7 (5.8%)
ORG-T	27.5 (–)	11.9 (–)	15.6 (–)	13.9 (–)

racy in all categories compared with *ORG-L*. Improvements in the DEFINITION category were especially remarkable unlike the results for the iUnit Ranking subtask. We executed sign tests and statistically significant differences between *ORG-L* and *ORG-T* in all categories were confirmed ($p < 0.05$ for the CELEBRITY category and $p < 0.01$ for LOCAL, DEFINITION, and QA categories).

5.3 Discussion

The Hypothesis: "Relevant descriptions, i.e., important iUnits are contained in elements highly scored by element-based retrieval" were likely to be true throughout the experiments, since our method where element-based retrieval was applied to MobileClick outperformed the comparison methods and document retrieval. Additionally, even our document retrieval approach exceeded the comparison methods.

This research is focused on MobileClick and cannot directly be applied to more general automatic summarization tasks because our method requires that iUnit, i.e., relevant text fragments are available and targeted documents are structured. In other words, our method is applicable to more general automatic summarization tasks by achieving automatic iUnits extraction and structuring documents. We expect that iUnits can be extracted from elements with highly scored as inferred by the Hypothesis. Regarding structuring documents, some researches attempt to structure a plain text to a structured document, such as XML [20].

6. Related Work

This section reviews methods for the baseline and top-ranked runs of MobileClick.

The baseline method [9] uses the odds ratio to calculate a term weight in an iUnit. Then, an iUnit score is calculated by using the sum of term weights of query keywords in the iUnit as follows:

$$OR(u) = \sum_{w \in u} \frac{P_q(w)}{P_o(w)} \tag{20}$$

$$P_q(w) = \frac{n_{D_q,w}}{n_{D_q}} \tag{21}$$

$$P_o(w) = \frac{n_{D_o,w}}{n_{D_o}} \tag{22}$$

where u is an iUnit for a query q, $P_q(w)$ is the probability of a word w in a document set retrieved by q, and $P_o(w)$ is the probability of w in document sets, $n_{D_q,w}$ is the frequency of w in a document set D_q retrieved by q, n_{D_q} is the number of words in D_q, $n_{D_o,w}$ is the frequency of w in a document set D_o retrieved by other than q, and n_{D_o} is the number of words in D_o. Intuitively, a term weight increases as the term frequently occurs in documents containing query keywords. Similarly, a term weight decreases as the term does not frequently occur in documents containing query keywords.

Lai et al. [15] proposed a latent Dirichlet allocation (LDA)-based approach where topics generated by LDA are regarded as intents and achieved the highest formal run results for the iUnit Ranking subtask. First, a LDA model is build with document set. Then, intent probability of each query is calculated with the LDA model. Specifically, a query is treated as a short document and latent topic distribution of the query is inferred. Supposed a query Q consists of r terms $(q_1, q_2, \ldots, q_r)$, the latent topic representation of an intent i of a query keyword $q_j (\in Q)$, $P(i|q_j)$, is calculated as follow:

$$P(i|q_j) = \frac{P(q_j|i)P(i)}{P(q_j)} \tag{23}$$

where $P(q_j|i)$ is inferred by the LDA model. Next, a latent topic is sampled n times for each q_j. Consequently, a total of nr latent topics are accumulated. Finally, the probability of the latent topic i given the query q, $P(i|q)$, is formulated:

$$P(i|q) = \frac{c_i}{nr} \tag{24}$$

where c_i is the number of latent topic i being sampled. After the probability of each latent topic, i.e., intent probability, is calculated, a topic is assigned to each document. $i*$ is the assigned topic of a document d if a topic probability of $i*$ of d is highest as follow:

$$i* = \arg\max_{i \in I} P(i|d) \tag{25}$$

where I is a set of latent topics. As a result, a document list for each topic was constructed. The next step is to rank documents by extracting a document from each topic in a round-robin manner. Once documents are extracted from all topics, extraction re-starts from the first topic until all documents are ranked. Finally, an iUnit score is calculated by the number of co-occurring terms of an iUnit and a document. Normalization by document rank order is then applied.

Chellal and Boughanem [2] proposed a term weighting scoring function that was based on Shannon entropy and expanded with term frequency, document frequency, and word2vec. Then, an iUnit score is calculated by a sum of weights of terms in the iUnit. Among some variations of their methods, they reported the best one is following formula.

$$Score(u, q) = -\sum_{w \in u} P(w|D_q) \log_2(P(w|D_q))(1 + fr_q(w)) \tag{26}$$

$$P(w|D_q) = \frac{fr_{D_q}(w)}{|D_q|} \tag{27}$$

where $Score(u, q)$ is an iUnit score of an iUnit u in terms of a query q, w is a term in u, D_q is a document set on q, $fr_q(w)$ is the number of occurrence of w in q, $fr_{D_q}(w)$ is the frequency of D_q, and $|D_q|$ is the number of words in D_q.

An appropriate procedure in iUnit ranking is conceivable where iUnits are not scored directly but scored with scored documents (elements for our method), since both Lai et al. [15] and our approaches achieved accurate search results. We interpreted that this is because iUnits with short lengths (many iUnits were composed of a few terms) are difficult to be assigned appropriate scores.

On the other hand, the reason why Chellal and Boughanem [2] attained higher accuracy than other odds ratio-based approaches is because they took into account term frequency and more complex statistics, unlike the odds ratio that just distinguishes whether or not a term occurs in a document.

7. Conclusion

We employed element-based retrieval to improve the accuracy of mobile information retrieval where two-layered search results are presented. The proposed method was first used to construct a ranked element list with element-based retrieval. Then, iUnits similar to highly ranked elements are ranked highly in the search results.

Methods of similarity calculations, decay function of rank order, and selection of elements used in iUnit scoring were proposed for the iUnit Ranking subtask. The best combinations in the proposed method outperformed state-of-the-art approaches. The intent ranking method to identify intents to gain more information and query candidates for the first and the second layers were investigated for the iUnit Summarization subtask. The proposed method statistically and significantly improved search accuracy by 9.7% compared with the baseline.

Automatic extraction of iUnits from elements is expected to be part of future work since this study exploited iUnits that were provided. Data formatting or logical structure extraction of Web documents could be another direction for future work.

Acknowledgments This work was partly supported by JSPS KAKENHI Grant Numbers 15K20990, 17K12684, 15H02701, 16H02908, 26280115, 26540042, and 25240014.

References

[1] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G.: Learning to Rank Using Gradient Descent, *Proc. 22th ICML*, pp.89–96 (2005).

[2] Chellal, A. and Boughanem, M.: IRIT at the NTCIR-12 MobileClick-2 Task, *Proc. 12th NTCIR*, pp.143–146 (2016).

[3] Dey, M., Mondal, A. and Das, D.: NTCIR-12 MOBILECLICK: Sense-based Ranking and Summarization of English Queries, *Proc. 12th NTCIR*, pp.138–142 (2016).

[4] Han, W.-B., Wang, H.-H. and Tsai, R.T.-H.: NCU IISR System for NTCIR-12 MobileClick2, *Proc. 12th NTCIR*, pp.115–117 (2016).

[5] Iizuka, S., Yumoto, T., Nii, M. and Kamiura, N.: UHYG at the NTCIR-12 MobileClick Task: Link-based Ranking on iUnit-Page Bipartite Graph, *Proc. 12th NTCIR*, pp.118–125 (2016).

[6] Kamps, J., Geva, S., Trotman, A., Woodley, A. and Koolen, M.: Overview of the INEX 2008 Ad Hoc Track, *Formal Proc. INEX 2008 Workshop*, pp.1–28 (2009).

[7] Kato, M.P., Ekstrand-Abueg, M., Pavlu, V., Sakai, T., Yamamoto, T. and Iwata, M.: Overview of the NTCIR-10 1CLICK-2 Task, *Proc. 10th NTCIR*, pp.182–211 (2013).

[8] Kato, M.P., Ekstrand-Abueg, M., Sakai, V.P.T., Yamamoto, T. and Iwata, M.: Overview of the NTCIR-11 MobileClick Task, *Proc. 11th NTCIR*, pp.195–207 (2014).

[9] Kato, M.P., Sakai, T., Yamamoto, T., Pavlu, V., Morita, H. and Fujita, S.: Overview of the NTCIR-12 MobileClick Task, *Proc. 12th NTCIR*, pp.104–114 (2016).

[10] Kazai, G., Lalmas, M. and de Vries, A.P.: The Overlap Problem in Content-Oriented XML Retrieval Evaluation, *Proc. 27th ACM SIGIR*, pp.72–79 (2004).

[11] Keyaki, A., Miyazaki, J. and Hatano, K.: Applying XML Element Retrieval Techniques to Web Documents, *DBSJ Journal*, Vol.13, No.1, pp.64–70 (2015).

[12] Keyaki, A., Miyazaki, J. and Hatano, K.: Element-based Retrieval@MobileClick-2, *Proc. 12th NTCIR*, pp.126–130 (2016).

[13] Keyaki, A., Miyazaki, J., Hatano, K., Yamamoto, G., Taketomi, T. and Kato, H.: A Path expression-Based Smoothing of Query Likelihood Model for XML Element Retrieval, *Proc. 1st ACIT*, pp.296–300 (2013).

[14] Keyaki, A., Miyazaki, J., Hatano, K., Yamamoto, G., Taketomi, T. and Kato, H.: Fast Incremental Indexing with Effective and Efficient Searching in XML Element Retrieval, *International Journal of Web Information Systems* (*IJWIS*), Vol.9, pp.142–164 (2013).

[15] Lai, K.P., Lam, W. and Bing, L.: CUIS at the NTCIR-12 MobileClick2 Task, *Proc. 12th NTCIR*, pp.134–137 (2016).

[16] Liu, F., Yu, C., Meng, W. and Chowdhury, A.: Effective Keyword search in Relational Databases, *Proc. of ACM SIGMOD*, pp.563–574 (2006).

[17] Liu, W., Robertson, S. and Macfarlane, A.: Field-Weighted XML Retrieval Based on BM25, *Formal Proc. INEX 2005 Workshop*, pp.161–171 (2006).

[18] Ogilvie, P. and Callan, J.: Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval, *Formal Proc. INEX 2005 Workshop*, pp.211–224 (2006).

[19] Ong, K., Chen, R.-C. and Scholer, F.: RMIT at the NTCIR-12 MobileClick-2: iUnit Ranking and Summarization Subtasks, *Proc. 12th NTCIR*, pp.104–114 (2016).

[20] Piao, Y., Wang, T. and Jiang, H.: From Text to XML by Structural Information Extraction, *Proc. IEEE International Conference on Computer and Communications* (*ICCC 2015*), pp.448–452 (2015).

[21] Sakai, T.: The Reliability of Metrics Based on Graded Relevance, *Journal of Information Processing and Management*, Vol.43, No.2, pp.531–548 (2007).

[22] Sakai, T. and Dou, Z.: Summaries, Ranked Retrieval and Sessions: A Unified Framework for Information Access Evaluation, *Proc. 36th SIGIR*, pp.473–482 (2013).

[23] Sakai, T., Dou, Z., Yamamoto, T., Liu, Y., Zhang, M. and Song, R.: Overview of the NTCIR-10 INTENT-2 Task, *Proc. 10th NTCIR*, pp.94–123 (2013).

[24] Sakai, T., Kato, M.P. and Song, Y.-I.: Overview of NTCIR-9 1CLICK, *Proc. 9th NTCIR*, pp.180–201 (2011).

[25] Tajima, K., Hatano, K., Matsukura, T., Sano, R. and Tanaka, K.: Discovery and Retrieval of Logical Information Units in Web, *Proc. WOWS in Conjunction with the 4th DL*, pp.180–201 (1999).

[26] Yamamoto, T., Liu, Y., Zhang, M., Dou, Z., Zhou, K., Markov, I., Kato, M.P., Ohshima, H. and Fujita, S.: Overview of the NTCIR-12 IMine-2 Task, *Proc. 12th NTCIR*, pp.8–26 (2016).

[27] Yoshioka, T. and Yukawa, T.: NUTKS at NTCIR-12 MobileClick2: iUnit Ranking Subtask Using Topic Model, *Proc. 12th NTCIR*, pp.131–133 (2016).

Atsushi Keyaki He is an assistant professor at the School of Computing, Tokyo Institute of Technology. He received his Ph.D. degree in engineering from the Nara Institute of Science and Technology (NAIST) in 2014. He was a JSPS research fellow (DC2) from 2012 to 2014, and a research intern at Microsoft Research Asia in 2013, and a visiting researcher at National University of Singapore (NUS) from 2016 to 2017. He is a member of ACM, IPSJ, IEICE, DBSJ, NLP, and JSAI.

Jun Miyazaki He joined the faculty of Department of Computer Science, Tokyo Institute of Technology in 2013, where he is currently a professor. Before joining Tokyo Tech., he served as an associate professor at Nara Institute of Science and Technology (NAIST). He is a member of ACM, IEEE Computer Society, IEICE, IPSJ, and HIS.

Kenji Hatano He is currently a professor at Faculty of Culture and Information Science, Doshisha University. Before joining Doshisha University, he served as a postdoctoral fellow at Kobe University, an assistant professor at Nara Institute of Science and Technology (NAIST), and a visiting scientist at AT&T-Labs Research. He is a member of ACM, IEEE Computer Society, IEICE, and IPSJ.

(Editor in Charge: *Takayuki Yumoto*)